

**LAPORAN PRAKTIKUM**  
**MODUL IV**  
**LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun oleh:**  
**Andika Indra Prastawa**  
**NIM: 2311102033**

**Dosen Pengampu:**

Wahyu Andi Saputra, S.Pd.,M.Eng.

**PROGRAM STUDI TEKNIK INFORMATIKA**  
**FAKULTAS INFORMATIKA**  
**INSTITUT TEKNOLOGI TELKOM PURWOKERTO**  
**PURWOKERTO**  
**2023**

# **BAB I**

## **TUJUAN PRAKTIKUM**

- A. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
- B. Praktikan dapat membuat linked list circular dan non circular.
- C. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

## BAB II

### DASAR TEORI

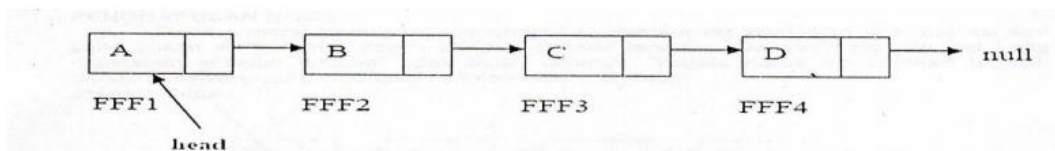
#### A. LINKED LIST NON CIRCULAR

linked list dapat diilustrasikan seperti rangkaian kereta api yang terdiri dari beberapa gerbong, di mana setiap gerbong adalah node dalam linked list. Agar node-node ini terhubung, kita memerlukan minimal satu pointer yang menghubungkan satu node ke node berikutnya dalam rangkaian. Setelah mendeklarasikan tipe data dan pointer untuk linked list, selanjutnya kita dapat membuat operasi-operasi dasar seperti penambahan, penghapusan, dan pengeditan node dalam linked list.

Berikut adalah contoh untuk mendeklarasikan single linked list non circular kedalam code cpp :

```
#include <iostream.h>
typedef struct Gerbong          //membuat sebuah tipe data baru yang terdiri dari
2 field
{
    int data;                  //field data bertipe integer
    Gerbong *next;             //field next merupakan pointer dari list
};
Gerbong *baru;                //variable baru bertipe pointer dari list
Gerbong *kepala;               //variable kepala bertipe pointer dari list
Gerbong *ekor;                 //variable ekor bertipe pointer dari list
Gerbong *bantu;                //variable bantu bertipe pointer dari list
Gerbong *hapus;                //variable hapus bertipe pointer dari list
```

Inti dari linked list adalah proses (tambah, edit, hapus) dari gerbong / node dan bagaimana menyambungkan antar gerbong / node tersebut.

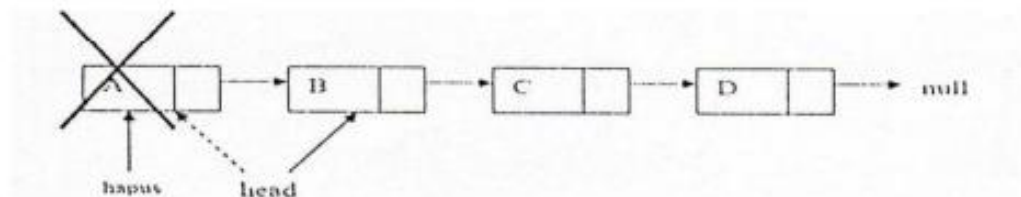


Gambar diatas diilustrasikan sebuah rangkaian kereta api dengan 4 buah gerbong. Gerbong A akan disebut sebagai kepala / head (walaupun penamaan ini bebas) dan gerbong D adalah ekor / tail. Tanda panah merupakan kait atau pointer yang

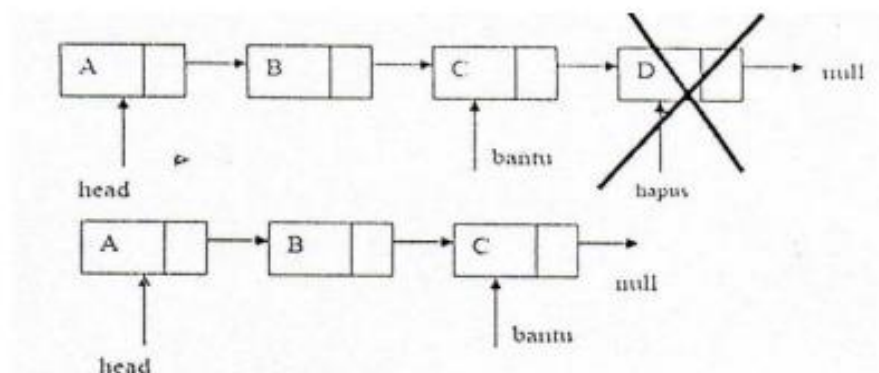
menghubungkan satu gerbong dengan yang lainnya. Pointer yang dimiliki D menuju ke NULL, inilah yang membedakan antara senarai berputar dengan yang tidak berputar. Kalau senarai berputar maka pointer dari D akan menuju ke A lagi. Pada senarai berkepala, penghapusan sebuah list dilakukan jika ada list lain yang bukan list "kepala" dalam barisan senarai tersebut. Dengan kata lain, list yang digunakan sebagai "kepala" tidak boleh dihapus, "kepala" harus dipindahkan terlebih dahulu. Keyword yang digunakan adalah delete.

Contohnya

### HAPUS DEPAN

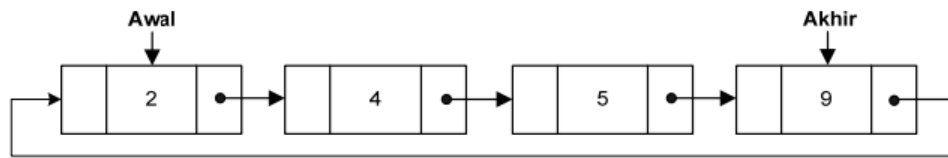


### HAPUS BELAKANG

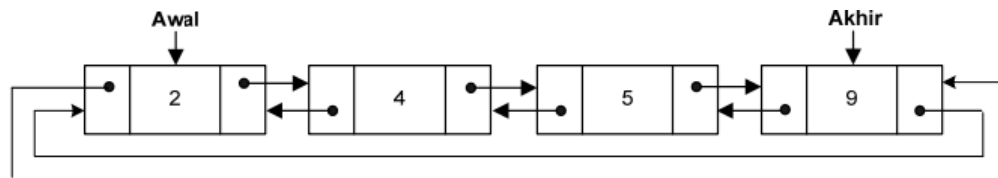


## B. LINKED LIST CIRCULAR

Senarai Berantai Melingkar adalah variasi dari senarai berantai di mana semua simpulnya terhubung, membentuk lingkaran. Ini berarti tidak ada NULL di bagian akhir. Node terakhir, alih-alih menunjuk ke NULL, menunjuk ke node pertama. Senarai berantai tunggal atau senarai berantai ganda dapat diubah menjadi senarai berantai



melingkar.



## BAB III

### GUIDED

#### 1. Guided 1

##### Source code

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST NON-CIRCULAR
// Deklarasi Struct Node
struct Node
{
    int data;
    Node *next;
};
Node *head;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
bool isEmpty()
{
    if (head == NULL)
        return true;
    else
        return false;
}
// Tambah Depan
void insertDepan(int nilai)
{
    // Buat Node baru
```

```
Node *baru = new Node;
baru->data = nilai;
baru->next = NULL;
if (isEmpty() == true)
{
    head = tail = baru;
    tail->next = NULL;
}
else
{
    baru->next = head;
    head = baru;
}
}
// Tambah Belakang
void insertBelakang(int nilai)
{
    // Buat Node baru
    Node *baru = new Node;
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        tail->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}
// Hitung Jumlah List
int hitungList()
```

```

{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah Tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi diluar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;
        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
    }
}

```



```

        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}
// Hapus Depan
void hapusDepan()
{
    Node *hapus;
    if (isEmpty() == false)
    {
        if (head->next != NULL)
        {
            hapus = head;
            head = head->next;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Belakang
void hapusBelakang()
{
    Node *hapus;
    Node *bantu;
    if (isEmpty() == false)
    {

```

```

        if (head != tail)
        {
            hapus = tail;
            bantu = head;
            while (bantu->next != tail)
            {
                bantu = bantu->next;
            }
            tail = bantu;
            tail->next = NULL;
            delete hapus;
        }
        else
        {
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}

// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
}

```

```

else
{
    int nomor = 1;
    bantu = head;
    while (nomor <= posisi)
    {
        if (nomor == posisi - 1)
        {
            sebelum = bantu;
        }
        if (nomor == posisi)
        {
            hapus = bantu;
        }
        bantu = bantu->next;
        nomor++;
    }
    sebelum->next = bantu;
    delete hapus;
}
}
// Ubah Depan
void ubahDepan(int data)
{
    if (isEmpty() == 0)
    {
        head->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
// Ubah Tengah

```

```

void ubahTengah(int data, int posisi)
{
    Node *bantu;
    if (isEmpty() == 0)
    {
        if (posisi < 1 || posisi > hitungList())
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)

```

```

    {
        tail->data = data;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

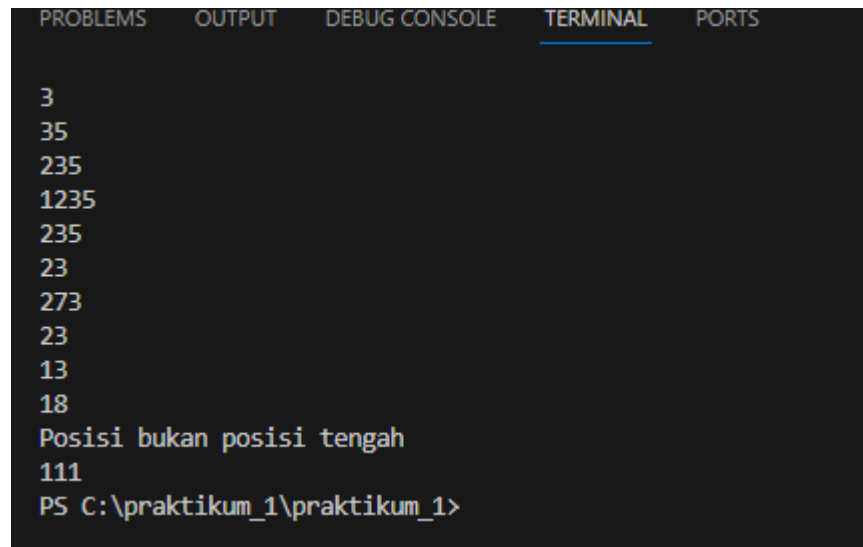
// Hapus List
void clearList()
{
    Node *bantu, *hapus;
    bantu = head;
    while (bantu != NULL)
    {
        hapus = bantu;
        bantu = bantu->next;
        delete hapus;
    }
    head = tail = NULL;
    cout << "List berhasil terhapus!" << endl;
}

// Tampilkan List
void tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
    }
}

```

```
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}
```

### Screenshoot program



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
3
35
235
1235
235
23
273
23
13
18
Posisi bukan posisi tengah
111
PS C:\praktikum_1\praktikum_1>
```

### Deskripsi program

Code diatas adalah implementasi dari Single Linked List non-circular. Struktur Linked List ini terdiri dari sejumlah node yang terhubung satu sama lain melalui pointer. Setiap node memiliki dua elemen utama: data untuk menyimpan nilai, dan pointer yang menunjuk ke node berikutnya. Kode menyediakan fungsi untuk operasi dasar seperti penambahan, penghapusan, dan pengubahan nilai node, serta tampilan isi Linked List. Melalui fungsi utama program, diberikan contoh penggunaan dari fungsi-fungsi tersebut. Ini adalah fondasi yang baik untuk memahami konsep Linked List.

## 2. Guided 2

### Source code

```
#include <iostream>
using namespace std;

// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};

Node *head = nullptr;
Node *tail = nullptr;
Node *baru = nullptr;
Node *bantu = nullptr;
Node *hapus = nullptr;

// Inisialisasi
void init()
{
    head = nullptr;
    tail = nullptr;
}

// Pengecekan apakah list kosong
bool isEmpty()
{
    return head == nullptr;
}
```



```
// Membuat node baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = nullptr;
}

// Menghitung jumlah node dalam list
int hitungList()
{
    int jumlah = 0;
    bantu = head;
    while (bantu != nullptr)
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}

// Menambah node di depan list
void insertDepan(string data)
{
    buatNode(data);
    if (isEmpty())
    {
        head = baru;
        tail = baru;
        baru->next = head;
    }
    else
    {

```

```

        baru->next = head;
        head = baru;
        tail->next = head;
    }
}

// Menambah node di belakang list
void insertBelakang(string data)
{
    buatNode(data);
    if (isEmpty())
    {
        head = baru;
        tail = baru;
        baru->next = head;
    }
    else
    {
        tail->next = baru;
        baru->next = head;
        tail = baru;
    }
}

// Menambah node di tengah list pada posisi tertentu
void insertTengah(string data, int posisi)
{
    if (isEmpty() || posisi == 1)
    {
        insertDepan(data);
    }
    else if (posisi > hitungList())
    {
        insertBelakang(data);
    }
}

```

```

    }
    else
    {
        baru = new Node;
        baru->data = data;
        bantu = head;
        for (int i = 1; i < posisi - 1; i++)
        {
            bantu = bantu->next;
        }
        baru->next = bantu->next;
        bantu->next = baru;
    }
}

// Menghapus node di depan list
void hapusDepan()
{
    if (!isEmpty())
    {
        hapus = head;
        if (head == tail)
        {
            head = nullptr;
            tail = nullptr;
        }
        else
        {
            head = head->next;
            tail->next = head;
        }
        delete hapus;
    }
    else

```

```

        {
            cout << "List masih kosong!" << endl;
        }
    }

// Menghapus node di belakang list
void hapusBelakang()
{
    if (!isEmpty())
    {
        hapus = head;
        if (head == tail)
        {
            head = nullptr;
            tail = nullptr;
        }
        else
        {
            bantu = nullptr;
            while (hapus->next != head)
            {
                bantu = hapus;
                hapus = hapus->next;
            }
            bantu->next = head;
            tail = bantu;
        }
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

```

```

// Menghapus node di tengah list pada posisi tertentu
void hapusTengah(int posisi)
{
    if (!isEmpty())
    {
        if (posisi == 1)
        {
            hapusDepan();
        }
        else if (posisi == hitungList())
        {
            hapusBelakang();
        }
        else
        {
            bantu = head;
            for (int i = 1; i < posisi - 1; i++)
            {
                bantu = bantu->next;
            }
            hapus = bantu->next;
            bantu->next = hapus->next;
            delete hapus;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Menghapus seluruh node dalam list
void clearList()

```

```
{  
    while (!isEmpty())  
    {  
        hapusDepan();  
    }  
    cout << "List berhasil terhapus!" << endl;  
}  
  
// Menampilkan isi list  
void tampil()  
{  
    if (!isEmpty())  
    {  
        bantu = head;  
        do  
        {  
            cout << bantu->data << " ";  
            bantu = bantu->next;  
        } while (bantu != head);  
        cout << endl;  
    }  
    else  
    {  
        cout << "List masih kosong!" << endl;  
    }  
}  
  
int main()  
{  
    init();  
    insertDepan("Ayam");  
    tampil();  
  
    insertDepan("Bebek");  
}
```

```
tampil();

insertBelakang("Cicak");
tampil();

insertBelakang("Domba");
tampil();

hapusBelakang();
tampil();

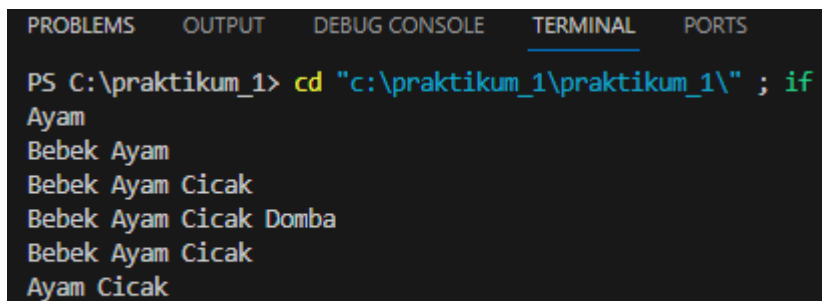
hapusDepan();
tampil();

insertTengah("Sapi", 2);
tampil();

hapusTengah(2);
tampil();

return 0;
}
```

### Screenshoot program



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\praktikum_1> cd "c:\praktikum_1\praktikum_1\" ; if
Ayam
Bebek Ayam
Bebek Ayam Cicak
Bebek Ayam Cicak Domba
Bebek Ayam Cicak
Ayam Cicak
```

**Deskripsi program**

Di atas adalah implementasi dari linked list berkaitan circular, yang termasuk struktur Node yang memuat data dan pointer ke node berikutnya, dan beberapa variabel global yang digunakan seluruh program. Program ini menyediakan fungsi untuk menginisialisasi daftar, menambah dan membuang node dari daftar, serta menampilkan isinya. Fungsi utama menunjukkan cara menggunakan fungsi-fungsi tersebut dengan membuat sebuah daftar yang berisi beberapa elemen string, dan melakukan operasi seperti menambah dan membuang node pada posisi yang berbeda, serta menampilkan daftar hasil setelah setiap operasi. Secara keseluruhan, program ini menjadi contoh sederhana tentang cara mengimplementasikan dan memanipulasi linked list berkaitan circular.



## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;

// Deklarasi Struct Node
struct Node
{
    string nama;
    string nim;
    Node *next;
};

// Pointer Global
Node *head = nullptr;

// Fungsi-fungsi
bool isEmpty()
{
    return head == nullptr;
}

void tambahDepan()
{
    string nama, nim;
    cout << "=====[TAMBAH DEPAN]=====" << endl;
    cout << "Masukkan Nama   : ";
    cin >> nama;
    cout << "Masukkan NIM      : ";
```

```

        cin >> nim;

        Node *baru_2311102033 = new Node;
        baru_2311102033->nama = nama;
        baru_2311102033->nim = nim;
        baru_2311102033->next = head;
        head = baru_2311102033;

        cout << "Data telah ditambahkan" << endl;
    }

void tambahBelakang()
{
    cout << "=====[TAMBAH BELAKANG]=====" << endl;
    string nama, nim;
    cout << "Masukkan Nama   : ";
    cin >> nama;
    cout << "Masukkan NIM     : ";
    cin >> nim;

    Node *baru_2311102033 = new Node;
    baru_2311102033->nama = nama;
    baru_2311102033->nim = nim;
    baru_2311102033->next = nullptr;

    if (isEmpty())
    {
        head = baru_2311102033;
    }
    else
    {
        Node *tail = head;
        while (tail->next)
        {

```

```

        tail = tail->next;

    }
    tail->next = baru_2311102033;
}

cout << "Data telah ditambahkan" << endl;
}

void tambahTengah()
{
    cout << "=====[TAMBAH TENGAH]=====" << endl;
    string nama, nim;
    int posisi;
    cout << "Masukkan Nama      : ";
    cin >> nama;
    cout << "Masukkan NIM       : ";
    cin >> nim;
    cout << "Masukkan Posisi    : ";
    cin >> posisi;

    if (posisi < 1)
    {
        cout << "Posisi harus lebih dari 0." << endl;
        return;
    }

    if (posisi == 1)
    {
        tambahDepan();
        return;
    }

    Node *current = head;
    for (int i = 1; i < posisi - 1; ++i)

```

```

    {
        if (!current)
        {
            cout << "Posisi melebihi panjang linked list." << endl;
            return;
        }
        current = current->next;
    }

    if (!current)
    {
        cout << "Posisi melebihi panjang linked list." << endl;
        return;
    }

    Node *baru_2311102033 = new Node;
    baru_2311102033->nama = nama;
    baru_2311102033->nim = nim;
    baru_2311102033->next = current->next;
    current->next = baru_2311102033;

    cout << "Data telah ditambahkan" << endl;
}

void ubahDepan()
{
    cout << "=====[UBAH DEPAN]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }
    string prev_nama = head->nama;

```

```

        string nama, nim;
        cout << "Masukkan Nama Baru : ";
        cin >> nama;
        cout << "Masukkan NIM Baru : ";
        cin >> nim;

        head->nama = nama;
        head->nim = nim;

        cout << "Data " << prev_nama << " telah diganti dengan data " <<
nama << endl;
    }

void ubahBelakang()
{
    cout << "=====[UBAH BELAKANG]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }

    Node *current = head;
    while (current->next)
    {
        current = current->next;
    }

    string nama, nim;
    cout << "Masukkan Nama Baru : ";
    cin >> nama;
    cout << "Masukkan NIM Baru : ";
    cin >> nim;

```

```

        string namaSebelum = current->nama;
        current->nama = nama;

        cout << "Data " << namaSebelum << " telah diganti dengan data " <<
nama << endl;
    }

void ubahTengah()
{
    cout << "=====[UBAH TENGAH]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }

    int posisi;
    cout << "Masukkan Posisi Data yang akan Diubah: ";
    cin >> posisi;

    Node *current = head;
    for (int i = 1; i < posisi; ++i)
    {
        if (!current)
        {
            cout << "Posisi melebihi panjang linked list." << endl;
            return;
        }
        current = current->next;
    }

    if (!current)
    {
        cout << "Posisi melebihi panjang linked list." << endl;
    }
}

```

```

        return;
    }

    string prev_nama = current->nama;
    string prev_nim = current->nim;

    string nama, nim;
    cout << "Masukkan Nama Baru : ";
    cin >> nama;
    cout << "Masukkan NIM Baru : ";
    cin >> nim;

    current->nama = nama;
    current->nim = nim;

    cout << "Data " << prev_nama << " telah diganti dengan data " <<
nama << endl;
}

void hapusDepan()
{
    cout << "=====[HAPUS DEPAN===== " << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }
    string namaHapus = head->nama;
    Node *hapus = head;
    head = head->next;
    delete hapus;

    cout << "Data " << namaHapus << " berhasil dihapus" << endl;
}

```

```

void hapusBelakang()
{
    cout << "=====[HAPUS BELAKANG]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }

    Node *current = head;
    Node *previous = nullptr;

    while (current->next)
    {
        previous = current;
        current = current->next;
    }

    if (previous)
    {
        previous->next = nullptr;
    }
    else
    {
        head = nullptr;
    }

    cout << "Data " << current->nama << " berhasil dihapus" << endl;

    delete current;
}

void hapusTengah()

```



```

{
    cout << "=====[HAPUS TENGAH]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }

    int posisi;
    cout << "Masukkan Posisi Data yang akan Dihapus: ";
    cin >> posisi;

    if (posisi < 1)
    {
        cout << "Posisi harus lebih dari 0." << endl;
        return;
    }

    if (posisi == 1)
    {
        hapusDepan();
        return;
    }

    Node *current = head;
    Node *previous = nullptr;
    int count = 1;
    while (current != nullptr && count < posisi)
    {
        previous = current;
        current = current->next;
        count++;
    }
}

```

```
        if (current == nullptr)
        {
            cout << "Posisi melebihi panjang linked list." << endl;
            return;
        }

        previous->next = current->next;

        cout << "Data " << current->nama << " berhasil dihapus" << endl;

        delete current;
    }

void hapusList()
{
    cout << "=====[HAPUS LIST]=====" << endl;
    while (!isEmpty())
    {
        hapusDepan();
    }
    cout << "List berhasil dihapus" << endl;
}

void tampilkanData()
{
    cout << "=====[TAMPILKAN DATA]=====" << endl;
    if (isEmpty())
    {
        cout << "List masih kosong." << endl;
        return;
    }
}
```

```

        cout << endl;
        "===== " << endl;
        cout << " | No |          NAMA          |          NIM          | " << endl;
        cout << endl;
        "===== " << endl;
        endl;

        Node *current = head;
        int no = 1;
        while (current)
        {
            cout << " | " << setw(3) << no << " | " << setw(15) << current->nama << " | " << setw(11) << current->nim << " | " << endl;
            current = current->next;
            no++;
        }
        cout << endl;
        "===== " << endl;
        endl;
    }

    int main()
    {
        int pilihan;
        do
        {
            cout << "=====| SELAMAT DATANG DI PROGRAM SINGLE LINKED LIST |===== " << endl;
            cout << endl;
            "===== " << endl;
            cout << " |          MENU          | " << endl;
        } while (pilihan != 0);
    }
}

```

```

        cout << endl;
        cout << "===== " << endl;
        cout << " [1]          Tambah Depan" << endl;
        cout << " [2]          Tambah Belakang" << endl;
        cout << " [3]          Tambah Tengah" << endl;
        cout << " [4]          Ubah Depan" << endl;
        cout << " [5]          Ubah Belakang" << endl;
        cout << " [6]          Ubah Tengah" << endl;
        cout << " [7]          Hapus Depan" << endl;
        cout << " [8]          Hapus Belakang" << endl;
        cout << " [9]          Hapus Tengah" << endl;
        cout << " [10]         Hapus List" << endl;
        cout << " [11]         Tampilkan" << endl;
        cout << " [12]         Keluar" << endl;
        cout << endl;
        cout << "\nMasukkan Pilihan Anda: ";
        cin >> pilihan;

        switch (pilihan)

```

```
{
case 1:
    tambahDepan();
    break;
case 2:
    tambahBelakang();
    break;
case 3:
    tambahTengah();
    break;
case 4:
    ubahDepan();
    break;
case 5:
    ubahBelakang();
    break;
case 6:
    ubahTengah();
    break;
case 7:
    hapusDepan();
    break;
case 8:
    hapusBelakang();
    break;
case 9:
    hapusTengah();
    break;
case 10:
    hapusList();
    break;
case 11:
    tampilkanData();
    break;
```

```

        case 12:
            cout << "Terima kasih telah menggunakan Program Single Linked
List Non-Circular!" << endl;
            cout << "Semoga harimu menyenangkan dan sukses selalu! " <<
endl;
            break;
        default:
            cout << "Pilihan tidak valid." << endl;
            break;
    }
} while (pilihan != 12);

return 0;
}

```

### Screenshoot program

```

=====| SELAMAT DATANG DI PROGRAM SINGLE LINKED LIST |=====
=====
|                                     MENU                                     |
=====
| [1]   Tambah Depan                  |
| [2]   Tambah Belakang              |
| [3]   Tambah Tengah                |
| [4]   Ubah Depan                   |
| [5]   Ubah Belakang                |
| [6]   Ubah Tengah                  |
| [7]   Hapus Depan                  |
| [8]   Hapus Belakang               |
| [9]   Hapus Tengah                 |
| [10]  Hapus List                   |
| [11]  Tampilkan                    |
| [12]  Keluar                       |
=====
Masukkan Pilihan Anda: |

```

### Tampilan operasi Tambah:

```

Masukkan Pilihan Anda: 1
=====[TAMBAH DEPAN]=====
Masukkan Nama   : andika
Masukkan NIM    : 2311102033
Data telah ditambahkan

```

```
Masukkan Pilihan Anda: 2
=====[TAMBAH BELAKANG]=====
Masukkan Nama : jkt
Masukkan NIM : 2319876654
Data telah ditambahkan
```

```
Masukkan Pilihan Anda: 3
=====[TAMBAH TENGAH]=====
Masukkan Nama : ijat
Masukkan NIM : 23144323232
Masukkan Posisi : 2
Data telah ditambahkan
```

#### Tampilan Operasi Ubah:

```
Masukkan Pilihan Anda: 4
=====[UBAH DEPAN]=====
Masukkan Nama Baru : indra
Masukkan NIM Baru : 2311102033
Data andika telah diganti dengan data indra
```

```
Masukkan Pilihan Anda: 5
=====[UBAH BELAKANG]=====
Masukkan Nama Baru : upin
Masukkan NIM Baru : 23119875432
Data jkt telah diganti dengan data upin
```

```
Masukkan Pilihan Anda: 6
=====[UBAH TENGAH]=====
Masukkan Posisi Data yang akan Diubah: 2
Masukkan Nama Baru : ipin
Masukkan NIM Baru : 23111765656
Data ijat telah diganti dengan data ipin
```

#### Tampilan Operasi Hapus:

```
Masukkan Pilihan Anda: 7
=====[HAPUS DEPAN]=====
Data indra berhasil dihapus
```

```
Masukkan Pilihan Anda: 8
=====[HAPUS BELAKANG]=====
Data upin berhasil dihapus
```

```

Masukkan Pilihan Anda: 9
=====[HAPUS TENGAH]=====
Masukkan Posisi Data yang akan Dihapus: 2
Data oki berhasil dihapus
===== SELAMAT DATANG DI PROGRAM SINGLE LINKED LIST =====

```

### Tampilkan Operasi Tampil Data

```

=====
| No |      NAMA      |      NIM      |
=====
|  1 |      ipin      | 23111765656   |
|  2 |      anis      | 23111543433   |
=====

```

- Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

```

Masukkan Pilihan Anda: 11
=====[TAMPILKAN DATA]=====
=====
| No |      NAMA      |      NIM      |
=====
|  1 |      Jawad     | 233000001     |
|  2 |      andika    | 2311102033    |
|  3 |      farrel    | 233000003     |
|  4 |      Denis     | 233000005     |
|  5 |      Anis      | 233000008     |
|  6 |      Bowo      | 233000015     |
|  7 |      Gahar     | 233000040     |
|  8 |      Udin      | 233000048     |
|  9 |      Ucok      | 233000050     |
| 10 |      Budi      | 233000099     |
=====

```

- Lakukan perintah berikut:
  - Tambahkan data berikut diantara Farrel dan Denis:  
**Wati 23300004**



```

Masukkan Pilihan Anda: 3
=====[TAMBAH TENGAH]=====
Masukkan Nama      : Wati
Masukkan NIM       : 2330004
Masukkan Posisi   : 4
Data telah ditambahkan

```

- Hapus data Denis

```

Masukkan Pilihan Anda: 9
=====[HAPUS TENGAH]=====
Masukkan Posisi Data yang akan Dihapus: 5
Data Denis berhasil dihapus

```

- Tambahkan Data berikut di awal:

**Owi 2330000**

```

Masukkan Pilihan Anda: 1
=====[TAMBAH DEPAN]=====
Masukkan Nama      : owi
Masukkan NIM       : 2330000
Data telah ditambahkan

```

- Tambahkan Data berikut di akhir:

**David 23300100**

```

Masukkan Pilihan Anda: 2
=====[TAMBAH BELAKANG]=====
Masukkan Nama      : David
Masukkan NIM       : 23300100
Data telah ditambahkan

```

- Ubah data Udin menjadi data berikut:

**Idin 23300045**

```

Masukkan Pilihan Anda: 6
=====[UBAH TENGAH]=====
Masukkan Posisi Data yang akan Diubah: 9
Masukkan Nama Baru : Idin
Masukkan NIM Baru  : 23300045
Data Udin telah diganti dengan data Idin

```

- Ubah data terakhir menjadi berikut:

### Lucy 23300101

```
Masukkan Pilihan Anda: 5
=====[UBAH BELAKANG]=====
Masukkan Nama Baru : Lucy
Masukkan NIM Baru : 23300101
Data David telah diganti dengan data Lucy
```

- Hapus data awal

```
Masukkan Pilihan Anda: 7
=====[HAPUS DEPAN]=====
Data owi berhasil dihapus
```

- Ubah data awal menjadi berikut:

### Bagas 23300002

```
Masukkan Pilihan Anda: 4
=====[UBAH DEPAN]=====
Masukkan Nama Baru : Bagas
Masukkan NIM Baru : 23300002
Data Jawad telah diganti dengan data Bagas
```

- Hapus data akhir

```
Masukkan Pilihan Anda: 8
=====[HAPUS BELAKANG]=====
Data Lucy berhasil dihapus
```

- Tampilkan seluruh data

```
=====
Masukkan Pilihan Anda: 11
```

```
=====[TAMPILKAN DATA]=====
```

```
=====
| No |      NAMA      |      NIM      |
=====
| 1  |      Bagas     |      2330002   |
| 2  |      andika    |      2311102033|
| 3  |      farrel    |      23300003  |
| 4  |      Wati      |      23300004  |
| 5  |      Anis      |      23300008  |
| 6  |      Bowo      |      23300015  |
| 7  |      Gahar     |      23300040  |
| 8  |      Idin      |      23300045  |
| 9  |      Ucok      |      23300050  |
| 10 |      Budi      |      23300099  |
=====
```

**Deskripsi program**

Program di atas merupakan sebuah aplikasi untuk mengelola data menggunakan struktur data linked list satu arah (single linked list). Dalam program ini, pengguna dapat menambahkan, mengubah, menghapus, dan menampilkan data sesuai dengan pilihan yang tersedia melalui menu interaktif. Setiap data disimpan dalam sebuah struktur bernama Node, yang memiliki atribut nama dan nim, serta pointer next yang menunjukkan ke node berikutnya dalam linked list. Program menyediakan berbagai fitur, seperti menambah data ke depan, belakang, atau tengah linked list, mengubah data pada posisi tertentu, serta menghapus data dari linked list.

## **BAB IV**

### **KESIMPULAN**

Praktikum tentang Linked List Circular dan Non-Circular merupakan pengalaman yang berharga dalam memahami dan mengimplementasikan dua jenis struktur data yang penting dalam pemrograman, yaitu linked list. Dalam praktikum ini, kami mempelajari konsep dasar dari kedua jenis linked list dan menerapkannya dalam bahasa pemrograman C++. Dalam Linked List Non-Circular, setiap elemen (node) memiliki pointer yang menunjuk ke node berikutnya dalam urutan linear. Hal ini memungkinkan untuk menambah dan menghapus elemen dengan mudah, serta melakukan traversing dari awal hingga akhir linked list.

## DAFTAR PUSTAKA

- [1] Repositori Unikom (Universitas Komputer Indonesia ) disusun oleh A Mukharil Bachtiar · 2010 Diakses  
<https://repository.unikom.ac.id/32762/1/Bab%20VII%20-%20Circular%20Linked%20List.pdf>
  
- [2] website geeksforgeeks di akses :  
<https://www.geeksforgeeks.org/circular-linked-list/>